

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 1

JC542 U.S. PTO

09/209162



12/10/98

```
1 on startMovie
2
3   global emG_passwordList, emG_userGroupList, emG_userGroup,
4   emG_userName, emG_registeredUsers, emG_msgNumber, emG_maildata,
5   emG_mode, emG_noSimulate, emG_mailFileList, emG_boxName
6
7   --- Register the "YAK YAK" text to speech xtra
8   -- register xtra "Yak", "XXXXXXXXXXXXXXXX"
9
10  -- VARIABLE LIST
11  --- emG_userName: Tracks current user by name
12  --- emG_msgNumber: Tracks if a message is new (empty) or old
13  (number)
14  --- emG_registeredUsers: Tracks users for to boxes in movies
15  --- emG_passwordList: List of passwords for user login:
16  [password:name]
17  --- emG_maildata: Message data list:
18  ---      #to, #from, #re, #date, #mimetype, #mbxName, #msgbody
19  -- NOT IMPLEMENTED -> #mbxName: now takes the place of #status -
20  eliminate case statement...
21  --- emG_mode: flag for message movies; #author, #display
22  --- emG_noSimulate: disable simulate Mode for message handler movies
23  --- emG_userGroupList: for testing rebus game
24  --- emG_userGroup: for testing rebus
25  --- emG_mailFileList: List of locations of mailfiles for each user:
26  ---      [uname:filename]
27  --- emG_boxName: a mailbox datastructure; used to pass mailboxes to
28  the mailbox movie
29
30  -- Install the menu
31  installMenu "main menu"
32
33  -- Clear all global variables
34
35  set emG_noSimulate = TRUE
36
37  --- Make sure the AddUsers button is not visible
38  set the visible of sprite 20 = FALSE
39
40  initSystemUsersData()
41  initializeUser()
42  initializeFields()
43  fillStudentName()
44  clearPassword()
45
46 end
47
48
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 2

```
49  on stopMovie
50
51      global instanceOfXtra, emG_passwordList, emG_userGroupList,
52      emG_userGroup, emG_userName, emG_msgNumber, emG_maildata, emG_mode
53
54      -- Clear all fields and global variables
55
56      put "" into field "addPass"
57      put "" into field "addUserGroup"
58      put "" into field "addName"
59      put "" into field "userList"
60      put "" into field "studentName"
61      put "" into field "studentUpName"
62      put "" into field "studentPass"
63
64      put "" into emG_userName
65      set emG_msgNumber = 0
66      set emG_registeredUsers = []
67      set emG_passwordList = [:]
68      set emG_userGroupList = [:]
69      set emG_maildata = [:]
70
71      set emG_userGroup = 0
72      set emG_mode = #empty
73
74      clearPassword()
75
76      -- empty the script used to read in mailboxes
77      set the scriptText of member 65 = ""
78
79      --- Make sure the AddUsers button is not visible
80      set the visible of sprite 20 = FALSE
81
82  end
83
84
85  -- score script 3 ss_goTheFrame
86
87  on exitFrame
88
89      go the frame
90
91  end
92
93
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 3

```
94 --- Modified 8-9-98. To include a mailfile location for each
95 --- user. Added global variable emG_mailFileList. Also changed
96 --- format of the users file to be comma delimited items. This
97 --- will avoid problem with spaces in full pathnames for
98 --- user mailbox files.
99
100 on initSystemUsersData
101   global emG_registeredUsers
102   global emG_passwordList, emG_userGroupList, emG_mailFileList
103
104   set emG_registeredUsers = []
105   set emG_passwordList = [:]
106   set emG_userGroupList = [:]
107   set emG_mailFileList = [:]
108
109   set usersData = readUsersFile()
110
111   put the number of lines of usersData into totalLines
112   repeat with i = 1 to totalLines
113
114     if line i of usersData = EMPTY then
115       nothing
116     else
117       set uname = item 1 of line i of usersData
118       set pw = item 2 of line i of usersData
119       set ugroup = value(item 3 of line i of usersData)
120       set mfile = item 4 of line i of usersData
121
122       add emG_registeredUsers, uname
123       addProp emG_passwordList, uname, pw
124       addProp emG_userGroupList, uname, ugroup
125       addProp emG_mailFileList, uname, mfile
126     end if
127
128   end repeat
129
130   sortRegisteredUsers()
131
132 end initSystemUsersData
133
134 -----
135 on initializeUser
136
137   global emG_userGroup, emG_userName
138   global emG_msgNumber, emG_maildata, emG_mode
139
140   put "" into emG_userName
141   set emG_msgNumber = 0
142   set emG_maildata = [:]
143   set emG_userGroup = 0
144   set emG_mode = #empty
145
146 end initializeMyGlobals
147
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 4

```
148 -----
149 -- Initialize formatting of all visible text fields
150 -- Should be called when movie starts
151
152 on initializeFields
153
154     -- SetTextInfo "StudentName", " ", "left", "arial", 14, "bold"
155     SetTextInfo "StudentUpName", "your username here ", "left", "arial",
156     14, "bold"
157     SetTextInfo "StudentPass", "", "left", "arial", 14, "bold"
158
159     put "" into field "addPass"
160     put "" into field "addUserGroup"
161     put "" into field "addName"
162     put "" into field "userList"
163
164     -- set the lineHeight of field "To" = 18
165     -- set the border of member "To" = 1
166     -- set the border of member "ToDown" = 1
167     -- set the margin of member "ToDown" to 8
168
169 end initializeFields
170
171 -----
172 -- THIS HANDLER FILLS THE STUDENT LOGON NAME FIELD
173 -- WITH THE CURRENT LIST OF STUDENT NAMES
174
175 on fillStudentName
176     global emG_registeredUsers
177
178     -- Clear the student name field for the kids' logon
179     put "" & RETURN into field "studentName"
180
181     repeat with uname in emG_registeredUsers
182
183         put uname & RETURN after field "studentName"
184
185     end repeat
186
187     -- Bring the field back to the top line
188     set the scrollTop of member "studentName" = 0
189
190 end
191
192 -----
193 -- For convenience of all the message handling movies
194 -- keep emG_registeredUsers in a special sorted order:
195 -- alphabetic with "administrator" at the end.
196
197 on sortRegisteredUsers
198     global emG_registeredUsers
199
200     -- fix up emG_registeredUsers in sorted order but
201     -- with "administrator" at the end
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 5

```
202
203     deleteOne(emG_registeredUsers, "administrator")
204     sort(emG_registeredUsers)
205     append(emG_registeredUsers, "administrator")
206
207 end sortRegisteredUsers
208 --mailbox handlers
209
210 -----
211 --- openMailbox starts the mailbox movie
212 --- because the call must be continued in emh_continue
213 --- it is necessary to use a global variable for the
214 --- mailbox name.
215
216 on openMbx boxName
217     global emG_boxName
218
219     set emG_BoxName = boxName
220
221     go to frame "movie"
222
223     -- since all sprites are automatically puppets in Dir 6.0
224     -- next should not be necessary
225     -- Take control of the sidebar buttons
226
227     puppetSprite 6, TRUE
228     puppetSprite 7, TRUE
229     puppetSprite 8, TRUE
230     puppetSprite 9, TRUE
231
232     set mbxMovie = window "mailbox.dir"
233     set the titleVisible of mbxMovie to FALSE
234     set the rect of mbxMovie = getMovieRect("mailbox")
235
236     open mbxMovie
237     set the name of mbxMovie to "childWindow"
238
239     tell window "childWindow"
240         -- next is a hack to get around Macromedia MIAW bug
241         -- see emh_continue for calls to real handlers
242         emc_startMeUp()
243
244     end tell
245
246     -- CONTINUES in emh_continue
247 end
248
249 -----
250 -- Read mailbox accepts a string that is the mailbox name
251 -- and returns a mailbox datastructure that is the
252 -- mailbox name and a list of the messages in that box
253
254 on readMailbox boxName
255     global emG_userName, emG_mailFileList
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 6

```
256 --
257 -- "inbox" : set bxstring = "#mbxName: #received"
258 -- "outbox" : set bxstring = "#mbxName: #sent"
259 -- "savebox" : set bxstring = "#mbxName: #saved"
260 -- "trashbox" : set bxstring = "#mbxName: #trashed"
261
262     set msgList = []
263
264
265     set mbxStruc = list(boxName, msgList)
266     set mailFileName = getProp(emG_mailFileList, emG_userName)
267
268     -- Start up Fileio Xtra
269     set instanceOfXtra = new(xtra "fileio")
270
271     -- Set up Fileio to read from users file
272     openFile(instanceOfXtra, mailFileName, 1)
273
274
275     -- If file users doesn't exist, create it and set it up for read
276     if status(instanceOfXtra) <> 0 then
277         createFile(instanceOfXtra, mailFileName)
278         openFile(instanceOfXtra, mailFileName, 1)
279     end if
280
281     -- Read what's currently in the file
282     set whatText = readFile(instanceOfXtra)
283
284     -- put msgs from appropriate box into the message list
285     -- this needs to be fixed after the mail file datastructure
286     -- is changed...
287
288     -- if value(#mbxname) <> 0 then
289     --     alert "Invalid mailbox name."
290     --     return(0)
291     -- end if
292
293     --OLD case statement
294     case boxname of
295         "inbox" : set bxstring = "#status: #received"
296         "outbox" : set bxstring = "#status: #sent"
297         "savebox" : set bxstring = "#status: #saved"
298         "trashbox" : set bxstring = "#status: #trashed"
299
300     otherwise:
301         alert "Invalid mailbox name."
302         return(0)
303     end case
304
305     -- inefficient to have to look for the "#status...string"
306     -- now is changed to value(#string) turning the string into a value,
307 as
308     -- Director has difficulties with strings w/in property lists...
309
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 7

```
310     repeat with i = 1 to the number of lines in whatText
311
312         if line i of whatText contains bxstring then
313             append(msgList, value(line i of whatText))
314         end if
315
316     end repeat
317
318
319     -- Close Fileio Xtra
320
321     closeFile(instanceOfXtra)
322
323     set instanceOfXtra = 0
324
325     return(mbxStruc)
326
327 end
328
329 on messageHandler msgStatus
330
331     global emG_userName, emG_maildata, emG_msgNumber, emG_mode,
332     emG_mailFileList
333
334     put "" into sendData
335
336     setProp emG_maildata, #status, msgStatus
337
338
339     -- Set up where to find the users mailfile
340     set whatFile = getProp(emG_mailFileList, emG_userName)
341
342
343     -- Start up Fileio Xtra
344     set instanceOfXtra = new(xtra "fileio")
345
346
347     -- Set up Fileio to read and write from/to users file
348     openFile(instanceOfXtra, whatFile, 0)
349
350
351     -- If file users doesn't exist, create it and set it up for
352     read/write
353     if status(instanceOfXtra) <> 0 then
354         createFile(instanceOfXtra, whatFile)
355         openFile(instanceOfXtra, whatFile, 0)
356     end if
357
358
359     -- Read what's currently in the file
360     set whatText = readFile(instanceOfXtra)
361
362
363     -- Add message to current user's mailbox
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 8

```
364 --- if it previously existed, then write over the old message
365 --- if not, add it to the bottom
366 --- Only messages with a status = #saved can be changed.
367
368 if emG_msgNumber <> 0 then
369   repeat with i = 1 to the number of lines in whatText
370     if i = emG_msgNumber then
371       put emG_maildata & RETURN after sendData
372     else if line i of whatText <> "" then
373       put line i of whatText & RETURN after sendData
374     end if
375   end repeat
376
377 else if emG_msgNumber = 0 then
378   put whatText into sendData
379   put emG_maildata & RETURN after sendData
380 end if
381
382
383 -- Put the cursor at the begining of the users file
384 setPosition(instanceOfXtra, 0)
385
386
387 -- Overwrite users file with updated list
388 writeString(instanceOfXtra, sendData)
389
390
391 -- Close Fileio Xtra
392
393 closeFile(instanceOfXtra)
394
395 set instanceOfXtra = 0
396
397
398 -- ON SEND, PUT IN OTHER CHILD'S MAILBOX, TOO
399
400 if msgStatus = #sent then
401
402   setaProp emG_maildata, #status, #received
403
404   put getaProp(emG_maildata, #to) into sendingTo
405
406   put "" into sendData
407
408
409   -- Set up where to find the users file
410   -- put the pathName & sendingTo into whatFile
411   set whatFile = getProp(emG_mailFileList, sendingTo)
412
413
414   -- Start up Fileio Xtra
415   set instanceOfXtra = new(xtra "fileio")
416
417
```


Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 9

```
418      -- Set up Fileio to read and write from/to users file
419      openFile(instanceOfXtra, whatFile, 0)
420
421
422      -- If file users doesn't exist, create it and set it up for
423      read/write
424      if status(instanceOfXtra) <> 0 then
425          createFile(instanceOfXtra, whatFile)
426          openFile(instanceOfXtra, whatFile, 0)
427      end if
428
429
430      -- Read what's currently in the file
431      set whatText = readFile(instanceOfXtra)
432
433      -- Add message to recipient's mailbox
434      put emG_maildata & RETURN after whatText
435
436      -- Put the cursor at the beginning of the users file
437      setPosition(instanceOfXtra, 0)
438
439      -- Overwrite users file with updated list
440      writeString(instanceOfXtra, whatText)
441
442
443      -- Close Fileio Xtra
444
445      closeFile(instanceOfXtra)
446      set instanceOfXtra = 0
447
448      end if
449
450      end
451      -----
452      on createMailData userName, type
453
454          set newmsg = [:]
455          addProp(newmsg, #to, "")
456          addProp(newmsg, #from, userName)
457          addProp(newmsg, #re, "")
458          addProp(newmsg, #date, the abbreviated date)
459          addProp(newmsg, #mimetype, type)
460          addProp(newmsg, #status, #new)
461          addProp(newmsg, #msgbody, [])
462          return(newmsg)
463
464      end createMailData
465
466      -----
467      --- Make sure there is something in each of the "to"
468      --- and "from" fields and that the messagebody has the
469      --- right format.
470
471      on isValidMessage maildata
```

Appendix A: KidCode@ Lingo Client/Server Email Main Scripts

Page 10

```
472
473     repeat with prop in [#to, #from]
474         if getProp(maildata, prop) = "" then
475             alert "But who do you wish to send this message to?"
476             --return(0)
477         end if
478     end repeat
479
480     if not listp(getProp(mailData, #msgBody)) then return(0)
481
482     return(1)
483
484 end isValidMessage
485
486 -----
487
488 on setReply
489
490     -- TAKES CARE OF SWITCHING THE SIDEBAR BUTTONS WHEN REPLY
491     -- IS HIT FROM AN OPEN MESSAGE
492
493     go to "Movie" -- make sure the frame is correct
494
495     -- Set the buttons with reply off and send on
496     disableReply()
497     enableSend()
498
499 end
500
501 -----
502
503 on disableSend
504     go to "movie"
505     puppetsprite 7, TRUE
506     set the member of sprite 7 = member "SendNo"
507 end disableSend
508
509 on enableSend
510     go to "movie"
511     puppetsprite 7, TRUE
512     set the member of sprite 7 = member "Send"
513 end enableSend
514
515 on disableReply
516     go to "movie"
517     puppetsprite 6, TRUE
518     set the member of sprite 6 = member "ReplyNo"
519 end disableSend
520
521 on enableReply
522     go to "movie"
523     puppetsprite 6, TRUE
524     set the member of sprite 6 = member "Reply"
```

Appendix A: KidCode@ Lingo Client/Server Email Main Scripts

Page 11

```
526 end enableSend
527
528
529 on returnToMain
530
531     global emG_msgNumber, emG_maildata, emG_mode
532
533     -- Clear the variables
534
535     set emG_msgNumber = 0
536     set emG_maildata = [:]
537     set emG_mode = #empty
538
539     --- unpuppet the left panel buttons which reuse sprite
540     --- channels 6-9
541     -- MB 10-13-98 I don't like this method... it is safer
542     -- to use new sprite channels.
543     -- is there a good reason for reusing channels...does it
544     -- affect performance?
545
546     puppetsprite 6, FALSE
547     puppetsprite 7, FALSE
548     puppetsprite 8, FALSE
549     puppetsprite 9, FALSE
550
551     -- Go back to the main menu
552
553     go to "open"
554
555 end
556
557 -- API handlers
558
559 --- emh_getUserMailbox returns the current user's mailbox specified
560 --- by the mailBoxName parameter.
561
562 on emh_getUserMailbox mailboxName
563
564     return(readMailbox(mailBoxName))
565
566 end emh_getUserMailbox
567
568 -----
569
570 on emh_getUserName
571     global emG_userName
572
573     return(emG_userName)
574
575 end emh_getUserName
576
577 -----
578
579 on emh_getUserData userName
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 12

```
580     global emG_userGroupList, emG_mailFileList
581
582     return(list (username, ~
583         username, getProp(emG_userGroupList, username),
584         getProp(emG_mailFileList, userName), [], list ( "inbox", "outbox",
585         "savebox") ))
586
587 end emh_getUserData
588
589
590 -- more API handlers
591
592 -----
593 --- A curse on Macromedia. This ugly hack is used to get
594 --- around a Macromedia bug which causes the startMovie
595 --- handler of a MIAW to run only after control has been
596 --- transferred back to the calling movie and the calling
597 --- movie advances a frame.
598
599 --- This handler is called by the startMovie handler of the
600 --- MIAW. This way we ensure that these scripts only run
601 --- after the MIAW has been properly initialized.
602
603 on emh_continue componentType
604     global emG_userName, emG_maildata, emG_mode, emG_boxName,
605     emG_userGroup
606
607     -- Since this function can only be called by a MIAW component
608     -- we assume that the "childwindow" is running
609
610     if componentType = #msgHandler then
611         tell window "childwindow"
612             emc_initWindow(emG_userName)
613             msh_openMessage(emG_maildata, emG_mode)
614         end tell
615
616     else if componentType = #mailbox then
617         tell window "childwindow" to emc_initWindow(emG_userName)
618         set success = the result
619         if not success then
620             alert "Could not initialize mailbox movie"
621             forget window "childwindow"
622             return(0)
623         end if
624
625         set mbx = readMailbox(emG_boxName)
626         tell window "childwindow" to mbx_openMailbox(mbx)
627         set success = the result
628         if not success then
629             alert "Could not open mailbox."
630             forget window "childwindow"
631             return(0)
632         end if
633     end if
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 13

```
634     else alert "ERROR invalid componenttype."
635
636 end emh_continue
637
638
639 -- more API handlers
640 -----
641 -- The emh_passMessage handler is used to pass a message from
642 -- a mailbox to the appropriate message handler
643
644 on emh_passMessage maildata, messageNumber
645
646     global emG_maildata, emG_msgNumber, emG_mode
647
648     -- should check for errors in the parameters
649
650     set emG_maildata = maildata
651     set emG_msgNumber = messageNumber
652
653     -- If a mailbox window is open we need to close that window.
654     -- The window will not actually close until this function completes
655     -- and returns control to the caller function in the mailbox movie.
656     -- Therefore, we need to move it to the back so it is no longer
657 visible.
658
659     moveToBack window "childwindow"
660     updateStage
661
662     tell window "childWindow" to emc_getComponentInfo()
663     set cInfo = the result
664     if getComponentProp(cInfo, #ComponentType) = #mailbox then
665         tell window "childWindow" to emc_closeWindow()
666         forget window "childWindow"
667     end if
668
669     go to frame "movie"
670     -- set up the button bar on the left
671
672     set msgStatus = getProp(emG_maildata, #status)
673     if msgStatus = #received then -- from inbox
674         set emG_mode = #display
675         disableSend()
676         enableReply()
677     else if msgStatus = #sent then -- from outbox
678         set emG_mode = #display
679         disableSend()
680         disableReply()
681     else if msgStatus = #saved then -- from savebox
682         set emG_mode = #author
683         disableReply()
684         enableSend()
685     else -- error
686         alert "passing message with invalid status"
687         return(0)
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 14

```
688     end if
689
690     --- OPEN MESSAGE HANDLER MOVIE
691
692     openMsgHandler(getaProp(emG_maildata,#mimetype), emG_maildata)
693
694 end emh_passMessage
695
696 -- more API handlers
697 -----
698 -- THIS CODE IS BASED ON OLD STUFF WHICH USES MESSAGE NUMBER
699 -- TO IDENTIFY MESSAGES ACROSS MAILBOXES. THIS SYSTEM
700 -- NEEDS TO BE CHANGED TO IDENTIFY MESSAGES BY A MAILBOXNAME
701 -- AND A MESSAGE NUMBER WITHIN THE BOX
702
703 on emh_getMessage messageNumber, typeorBoxName
704
705     global emG_userName, emG_msgNumber, emG_mailData
706
707     set emG_msgNumber = messageNumber
708
709     if messageNumber = 0 then -- return new message data
710         --typeorBoxName should have mimetype
711         set emG_maildata = createMailData(emG_userName, typeorBoxName)
712         return(emG_maildata)
713     end if
714
715     -- otherwise find an existing message
716     -- typeorboxname should have boxName
717
718     set theBox = readMailbox(typeorBoxName)
719     set emG_mailData = getaT(getAt(theBox, 2), messageNumber)
720     return(emG_maildata)
721
722 end emh_getMessage
723
724 -----
725
726 on emh_getRegisteredUsers
727     global emG_registeredUsers
728
729     return(emG_registeredUsers)
730
731 end emh_getRegisteredUsers
732
733 -----
734
735 on emh_killComponent
736
737     tell window "childwindow" to emc_closeWindow()
738     if the result = 0 then alert "TROUBLE CLOSING WINDOW!"
739     else
740         forget window "childwindow"
741     end if
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 15

```
742     returnToMain()
743
744 end emh_killComponent
745
746 -----
747 --- Initialize formatting of text fields
748 --- Thanks to Frank Leahy, maricopa site for this one
749
750 on SetTextInfo fldName, fldValue, fldAlign, fldFont, fldSize,
751 fldStyle
752
753     put fldValue into field fldName
754     set the textAlign of field fldName = fldAlign
755     set the textFont of field fldName = "arial"    --fldFont
756     set the textSize of field fldName = fldSize
757     set the textStyle of field fldName = fldStyle
758
759 end
760
761 -----
762
763 -- script of cast member studentName
764 -- emG_username should not be set here
765 -- because it could be invalid
766
767 on mouseUp
768
769     -- Put selected user name into up version of student field
770     -- switch the field from down to up
771
772     put word 1 of line(the mouseLine) of field "studentName" into field
773 "studentUpName"
774
775     set the member of sprite 14 to member "StudentUpName"
776
777 end
778
779 -----
780 -- script of cast member studentUpName
781
782 on mouseUp
783
784     -- Pull down student field: change field from
785     -- up (sprite 17) to down (sprite 16)
786
787     set the member of sprite 14 to member "StudentName"
788
789     -- clear password field
790     clearPassword()
791
792 end
793
794
795
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 16

```
796 -- scripts of cast member studentPassword
797
798
799 on keyUp
800     global gpw, gpwlen
801     --gpw is global password and
802     --gpwlen is global password length
803
804     hideAlert() -- user maybe trying again...hide badPwMsg
805
806     if the key = RETURN then
807         if checkPassword(field "studentUpName", gpw) then
808             enterMainEmail(field "studentUpName")
809         else --- invalid password
810             alertBadPassword()
811         end if
812         set gpw = ""
813         set gpwlen = 0
814         put "" into field "studentPass" -- reset the password field
815     end if
816
817 end keyUp
818
819 -----
820
821 on keyDown
822     global gpwlen, gpw
823
824     --eats the key, otherwise it will appear until keyup
825
826     if the key = BACKSPACE then
827         put "" into char gpwlen of field "studentPass"
828         put "" into char gpwlen of gpw
829         if gpwlen > 0 then
830             set gpwlen = gpwlen - 1
831         end if
832     else if the key = RETURN then
833         nothing
834     else if the keycode >= 117 and the keycode <= 126 then
835         nothing
836     else
837         put "*" after field "studentPass"
838         put the key after gpw
839         set gpwlen = gpwlen + 1
840
841     end if
842
843     set the selstart = gpwlen
844     set the selend = the selstart
845
846 end keyDown
847
848
849 -- script of cast member goStudentLog
```


Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 17

```
850
851 on mouseUp
852
853     go to frame "pass"
854
855 end
856
857 

---


858 -- script of cast member editUsers
859
860 on mouseUp
861
862     -- set the default pathname for the mail file location
863     put the pathname into field "addMailFileLoc"
864
865     go to frame "edit"
866
867 end
868
869
870 -- script of cast member okUser
871
872
873 on mouseDown
874     set the member of sprite 7 = "okay down"
875 end
876
877 

---


878
879 -- script of cast member okDown
880
881
882 on mouseUp
883     global gpw, gpwlen    --- see script of field studentPass
884
885     set the member of sprite 7 = "okayUser"
886
887     if checkPassword(field "studentUpName", gpw) then
888         -- valid user & pw
889         enterMainEmail(field "studentUpName")
890
891     else    -- password invalid
892
893         alertBadPassword()
894
895     end if
896
897     clearPassword()
898
899 end
```

Appendix A: KidCode@ Lingo Client/Server Email Main Scripts

Page 18

```
900 -- script of cast member addUser
901
902 on mouseUp
903     global emG_registeredUsers
904     global emG_passwordList, emG_userGroupList, emG_mailFileList
905
906     --check that username is filled and is unique
907
908     if field "addName" = EMPTY then
909         alert "No username"
910         return(0)
911     else if getOne(emG_registeredUsers, field "addName") then
912         alert "Username already in system. Choose a different name"
913         return(0)
914
915     else set uname = field "addName"
916
917     --NEED TO TAKE CARE OF THIS!!!!
918     -- check that the mailfile location is a valid directory
919     -- there are serious problems with this at present
920     -- for now assume pathnames are valid
921
922
923     -- add new User data to system global variables
924     add(emG_registeredUsers, uname)
925     addProp(emG_passwordList, uname, field "addPass")
926     addProp(emG_userGroupList, uname, field "addUserGroup")
927     -- append username to the mailfile location directory
928     addProp(emG_mailFileList, uname, field "addMailFileLoc" & uname)
929
930     sortRegisteredUsers()
931
932     -- write the users file with system users data
933     writeUsersFile()
934
935     -- Put the updated user list into the userList field
936     put "" into field "userList"
937     repeat with uname in emG_registeredUsers
938         put uname after field "userList"
939         put " " & getProp(emG_passwordList, uname) after field "userList"
940         put " " & getProp(emG_userGroupList, uname) after field "userList"
941         put " " & getProp(emG_mailFileList, uname) after field "userList"
942         put RETURN after field "userList"
943     end repeat
944
945
946     -- reset the User data fields
947
948     put "" into field "addUserGroup"
949     put "" into field "addPass"
950     put "" into field "addName"
951     put the pathname into field "addMailFileLoc"
952
953     -- Refill the kids' logon name field
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 19

```
954     fillStudentName()
955
956 end
957
958 -- script of cast member seeUserList
959
960 on mouseUp
961
962     global instanceOfXtra
963
964
965     put "" into field "userList"
966
967
968     -- Set up where to find the users file
969     put the pathName & "users" into whatFile
970
971
972     -- Start up Fileio Xtra
973     set instanceOfXtra = new(xtra "fileio")
974
975
976     -- Set up Fileio to read from users file
977     openFile(instanceOfXtra, whatFile, 1)
978
979
980     -- If file users doesn't exist, create it and set it up for read to
981     avoid error
982
983     if status(instanceOfXtra) <> 1 then
984         createFile(instanceOfXtra, whatFile)
985         openFile(instanceOfXtra, whatFile, 1)
986     end if
987
988
989     -- Read what's currently in the file
990     set whatText = readFile(instanceOfXtra)
991
992
993     -- Put the updated user list into the userList field
994     put whatText into field "userList"
995
996     -- Close Fileio Xtra
997     closeFile(instanceOfXtra)
998
999     set instanceOfXtra = 0
1000
1001 end
1002
1003
1004
1005 -- script of cast member DoneAdmin
1006
1007 on mouseUp
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 20

```
1008
1009     go to frame "open"
1010
1011     put "" into field "addName"
1012     put "" into field "addUserGroup"
1013     put "" into field "addPass"
1014     put "" into field "addMailFileLoc"
1015
1016 end
1017 -- msgHandlers scripts
1018 -----
1019 --- openMsgHandler starts the appropriate Message Handling movie.
1020 --- The call must be continued in emh_continue.
1021 --- It is necessary that the global variable emG_mailData is
1022 --- set up. Therefore, we pass it as a parameter to make it
1023 --- clear that the variable is necessary.
1024
1025 on openMsgHandler mimeType, mailData
1026
1027     set movieName = getMessageHandler(mimeType)
1028     go to frame "movie"
1029
1030     -- since all sprites are automatically puppets in Dir 6.0
1031     -- next should not be necessary
1032     -- Take control of the sidebar buttons
1033
1034     puppetSprite 6, TRUE
1035     puppetSprite 7, TRUE
1036     puppetSprite 8, TRUE
1037     puppetSprite 9, TRUE
1038
1039     set mshMovie = window movieName
1040     set the titleVisible of mshMovie to FALSE
1041     set the rect of mshMovie = getMovieRect(mimeType)
1042
1043     open mshMovie
1044     set the name of mshMovie to "childWindow"
1045
1046     tell window "childWindow"
1047         -- next is a hack to get around Macromedia MIAW bug
1048         -- see emh_continue for calls to real handlers
1049         emc_startMeUp()
1050
1051     end tell
1052
1053     -- CONTINUES in emh_continue
1054 end openMsgHandler
1055
1056 -----
1057
1058 -- getMessageHandler returns filename of movie to handle mimeType.
1059 -- This code makes it easy to make changes in movie filenames
1060 -- and to add new message handling movies.
1061
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 21

```
1062 on getMessageHandler mimetype
1063
1064     case mimetype of
1065         "text": return("text.dir")
1066         "rebus": return("rebus.dir")
1067         "grid": return("grid.dir")
1068         "connect": return("connect.dir")
1069         "puzzle" : return("puzzle.dir")
1070
1071     otherwise:
1072         alert "Invalid mimetype of message."
1073         return("")
1074     end case
1075
1076 end getMessageHandler
1077 -----
1078
1079 on getMovieRect whichMovie
1080
1081     --- the top of green panel
1082     set movieTop = the top of sprite 3
1083     --- the left of green panel
1084     set movieLeft = the left of sprite 3
1085
1086     case whichMovie of
1087         "rebus", "rebus.dir":
1088             set theRect= rect(movieLeft, movieTop, ~
1089                             movieLeft + 640, movieTop +480)
1090         "text", "text.dir":
1091             set theRect= rect(movieLeft, movieTop, ~
1092                             the stageRight - 5, the stageBottom -5)
1093         "puzzle", "puzzle.dir":
1094             set theRect= rect(movieLeft, movieTop, ~
1095                             the stageRight - 5, the stageBottom -5)
1096         "grid", "grid.dir", "connect", "connect.dir":
1097             set theRect= rect(movieLeft, movieTop, ~
1098                             the stageRight - 5, the stageBottom -5)
1099         "mailbox", "mailbox.dir":
1100             set theRect= rect(movieLeft, movieTop, ~
1101                             the stageRight - 5, the stageBottom -5)
1102     otherwise:
1103         alert "ERROR: invalid movieName: " & whichMovie
1104         set theRect = rect(0,0,0,0)
1105
1106     end case
1107
1108     return(theRect)
1109
1110 end getMovieRect
1111
1112
1113
1114 -- score script    fr_installMenu
1115
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 22

```
1116 on prepareFrame
1117   --first clear away any old menus
1118   installMenu 0
1119   installMenu "main menu"
1120 end
1121
1122
1123 -- password verification and user init
1124
1125
1126
1127 on enterMainEmail username
1128   global emG_username, emG_userGroup, emG_userGroupList
1129
1130   set emG_username = username
1131   set emG_userGroup = getProp(emG_userGroupList, emG_username)
1132
1133   -- ADMINISTRATOR has access to the "Edit Users" button
1134   if emG_username = "administrator" then
1135     set the visible of sprite 20 = TRUE
1136   end if
1137
1138   go to frame "open"
1139
1140 end enterMainEmail
1141
1142 -----
1143
1144
1145 on checkUserName userName
1146   global emG_registeredUsers
1147
1148   if getone(emG_registeredUsers, userName) then
1149     return(1) -- username is in system
1150
1151   else
1152     alert "User " & userName & "not a KidCode authorized user." & RETURN
1153     & "You cannot login without a valid user name."
1154
1155   end if
1156
1157 end checkUsername
1158
1159 -- more password handling scripts
1160
1161 on checkPassword userName, password
1162   global emG_passwordList
1163
1164   -- if the username is not valid quit this...
1165   if not checkUserName(userName) then return(0)
1166
1167   -- username is valid
1168
1169   -- First part of loop changes capital letters to lower case
```

Appendix A: KidCode@ Lingo Client/Server Email Main Scripts

Page 23

```
1170      -- Second part puts lower case letters into password check
1171      -- This eliminates all spaces and/or unacceptable characters
1172
1173      set checkPassword = ""
1174      repeat with i = 1 to the number of chars in password
1175
1176          put char(i) of password into capital
1177          put charToNum(capital) into capital
1178
1179          if capital <= 90 and capital >= 65 then
1180              put numToChar(capital + 32) after checkPassword
1181          else if capital >= 97 and capital <= 122 then
1182              put numToChar(capital) after checkPassword
1183          end if
1184
1185      end repeat
1186
1187      -- CHECK PASSWORD
1188
1189      set realPassword = getProp(emG_passwordList, username)
1190
1191      if realpassword = checkPassword then
1192          return(1)  --TRUE
1193      else
1194          return(0)
1195      end if
1196
1197
1198  end checkPassword
1199
1200  -----
1201
1202  on clearPassword
1203      global gpw, gpwlen
1204
1205      set gpw = ""
1206      set gpwlen = 0
1207      put "" into field "StudentPass"
1208
1209  end clearPassword
1210
1211  -----
1212
1213  on alertBadPassword
1214
1215      set the loc of sprite 17 to point(231, 350)
1216      beep()
1217
1218  end alertBadPassword
1219  on hideAlert
1220
1221      set the loc of sprite 17 to point(-188, -31)
1222
1223  end hideAlert
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 24

```
1224
1225
1226 -- script of cast member reply
1227
1228 on mouseUp
1229     global emG_userName, emG_msgNumber
1230     global emG_maildata, emG_mode, emG_userGroup
1231
1232     -- abandon current MailData which should be in the inbox.
1233     -- Later, the user may choose to either abandon or send
1234     -- the new replyTo message. That is not a concern.
1235
1236     -- If a mailbox window is open need to get the message
1237     -- and close that window.
1238
1239     tell window "childWindow" to emc_getComponentInfo()
1240     set cInfo = the result
1241     if getComponentProp(cInfo, #ComponentType) = #mailbox then
1242         tell window "childwindow" to mbx_GetMessageNumber()
1243         set emG_msgNumber = the result
1244         if emG_msgNumber <= 0 then
1245             alert "You must select a message."
1246             return() -- abandon the request to reply
1247         end if
1248
1249         tell window "childwindow" to mbx_GetMessage(emG_msgNumber)
1250         set emG_maildata = the result
1251
1252         --- forget window "childwindow" -- done in passMessage
1253
1254         --- Now open the appropriate Message Handler
1255         --- to display the message
1256
1257         emh_PassMessage(emG_maildata, emG_msgNumber)
1258
1259     end if
1260
1261     -- If we got to this point message handler is open.
1262     -- Presumably it has a message displayed. If the message
1263     -- is empty only the message handler knows that and it
1264     -- will need to catch the error and return an error code
1265     -- to msh_replyMessage.
1266
1267     -- The message handling movie's replyMessage handler
1268     -- should swap "to" and "from"
1269     -- fields and make the message editable
1270
1271     -- set mode to author to keep it consistent with msg handler
1272     set emG_mode = #author
1273
1274     set emG_msgNumber = 0 -- this is now a new message
1275
1276     tell window "childWindow"
1277         global emG_userGroup
```


Appendix A: KidCode@ Lingo Client/Server Email Main Scripts

Page 25

```
1278      -- msg handler will swap "to" with "from" and change
1279      -- mode to author
1280      moveToFront window "childWindow"
1281      msh_replyMessage()
1282  end tell
1283
1284  set emG_maildata = the result
1285
1286  -- Toggle the send and reply buttons
1287  setreply  -- disable reply and enable send buttons
1288
1289
1290 end
1291
1292
1293 -- script of cast member send
1294
1295 on mouseUp
1296   global emG_maildata, emG_userGroup
1297
1298   -- Could check that the childwindow is a messagehandler
1299   -- but this may not be necessary.
1300
1301   tell window "childWindow"
1302     global emG_userGroup
1303     msh_sendMessage()
1304     set emG_maildata = the result
1305   end tell
1306
1307   if not isValidMessage(emG_maildata) then
1308     alert "ERROR not a valid message."
1309     return(0)      -- abandon attempt to send
1310   end if
1311
1312   --- otherwise continue to send message
1313
1314   -- NEED TO FIX THIS SO THAT MESSAGE STATUS DOES NOT
1315   -- BECOME "#sent" if it fails to be saved to both
1316   -- mail files
1317
1318   messageHandler(#sent)  -- for now this uses global emG_maildata
1319
1320   -- tell window "childWindow" to msh_clearMessage()
1321
1322 end
1323
1324 -- script of cast member print
1325
1326 on mouseUp
1327
1328   tell window "childwindow" to emc_getComponentInfo()
1329   set cInfo = the result
1330   set cType = getComponentProp(cInfo, #ComponentType)
1331
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 26

```
1332     if cType = #mailbox then
1333         -- need to pass the message to its message handling
1334         -- component for printing. Ideally this can be done
1335         -- without opening a window and laying out the message.
1336
1337         alert "I can't do that right now. Open the message and then print."
1338
1339     else if cType = #msgHandler then
1340
1341         tell window "childwindow"
1342             msh_PrintMessage()
1343         end tell
1344
1345     else alert "ERROR invalid componenttype."
1346
1347 end
1348
1349
1350 -- script of cast member Quit
1351
1352 on mouseUp
1353
1354     handleQuit()
1355
1356 end
1357


---


1358
1359
1360 on handleQuit
1361
1362     initializeUser()
1363     clearPassword()
1364     go to frame 2
1365
1366     -- make sure the editUsers button is invisible
1367     set the visible of sprite 20 = FALSE
1368
1369 end handleQuit
1370
1371 -- script of cast member trash
1372
1373 --- Email Main now handles all aspects of trashing a
1374 --- message by writing the mail files. The components
1375 --- are instructed to update their state by clearing the
1376 --- message (if the component is a message handler) or
1377 --- redrawing the message list (if the component is a
1378 --- mailbox.)
1379
1380 --- Should add a confirmation dialog with the user
1381
1382 on mouseUp
1383     global emG_msgNumber -- number of the current message
1384
1385     tell window "childwindow" to emc_getComponentInfo()
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 27

```
1386     set cInfo = the result
1387     set cType = getComponentProp(cInfo, #ComponentType)
1388
1389
1390     if cType = #mailbox then
1391         -- need to determine which message(s) are currently
1392         -- selected and instruct the mailbox to update its
1393         -- display
1394
1395         -- temporary implementation of mbx_trashMessages does
1396         -- not handle multiple messages as a result the
1397         -- arguments are ignore...
1398
1399         tell window "childwindow" to mbx_trashMessages([])
1400
1401         -- the following lines will be necessary when
1402         -- mbx_trashMessages is properly implemented. For
1403         -- now, the temporary implementation trashes the
1404         -- message itself.
1405         -- set messageNumbers = the result
1406         -- delete each message in the list of messageNumbers
1407
1408
1409     else if cType = #msgHandler then
1410
1411         -- rewrite the message into the mailfile
1412         messageHandler(#trash)
1413
1414         tell window "childwindow" to msh_clearMessage()
1415
1416     else alert "ERROR invalid componenttype."
1417
1418
1419 end
1420
1421
1422 -- script of cast member text
1423
1424 on mouseUp
1425     global emG_msgNumber
1426     global emG_maildata, emG_mode
1427
1428     -- START A NEW MESSAGE
1429
1430     set emG_msgNumber = 0
1431     set emG_mode = #author
1432     set emG_maildata = createMailData(emG_userName, "text")
1433
1434     openMsgHandler("text", emG_mailData)
1435
1436     disableReply()
1437
1438 end
1439
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 28

```
1440 -----
1441 ---
1442 -- script of cast member Rebus
1443
1444 on mouseUp
1445     global emG_msgNumber
1446     global emG_maildata, emG_mode
1447
1448     -- START A NEW MESSAGE
1449
1450     set emG_msgNumber = 0
1451     set emG_mode = #author
1452     set emG_maildata = createMailData(emG_userName, "rebus")
1453
1454     openMsgHandler("rebus", emG_mailData)
1455
1456     disableReply()
1457
1458 end
1459
1460
1461 --- script of cast member grid
1462
1463 on mouseUp
1464     global emG_msgNumber
1465     global emG_maildata, emG_mode
1466
1467     -- START A NEW MESSAGE
1468
1469     set emG_msgNumber = 0
1470     set emG_mode = #author
1471     set emG_maildata = createMailData(emG_userName, "grid")
1472
1473     openMsgHandler("grid", emG_mailData)
1474
1475     disableReply()
1476
1477 end
1478
1479
1480 --- script of cast member puzzle
1481
1482 on mouseUp
1483     global emG_msgNumber
1484     global emG_maildata, emG_mode
1485
1486     -- START A NEW MESSAGE
1487
1488     set emG_msgNumber = 0
1489     set emG_mode = #author
1490     set emG_maildata = createMailData(emG_userName, "puzzle")
1491
1492     openMsgHandler("puzzle", emG_mailData)
1493
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 29

```
1494     disableReply()
1495
1496 end
1497
1498
1499 --- script of cast member connect
1500
1501 on mouseUp
1502     global emG_msgNumber
1503     global emG_maildata, emG_mode
1504
1505     -- START A NEW MESSAGE
1506
1507     set emG_msgNumber = 0
1508     set emG_mode = #author
1509     set emG_maildata = createMailData(emG_userName, "connect")
1510
1511     openMsgHandler("connect", emG_mailData)
1512
1513     disableReply()
1514
1515 end
1516
1517


---


1518 on GetComponentProp infoList, prop
1519
1520     --- need to add error checking code
1521
1522     case prop of
1523         #componentName: return(getAt(infolist, 1))
1524         #componentID: return(getAt(infolist, 2))
1525         #componentType: return(getAt(infolist, 3))
1526         #componentMIMETYPE: return(getAt(infolist, 4))
1527
1528         otherwise: alert "ERROR no component property."
1529     end case
1530
1531 end GetComponentProp
1532
1533 -- script of cast member savebox
1534
1535 on mouseUp
1536
1537     openMbx("savebox")
1538
1539 end
1540
1541
1542 -- script of cast member inbox
1543
1544 on mouseUp
1545
1546     openMbx("inbox")
1547
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 30

```
1548 end
1549
1550
1551
1552 -- script of cast member outbox
1553
1554 on mouseUp
1555
1556     openMbx("outbox")
1557
1558 end
1559
1560
1561 --- Users File functions
1562
1563 -- returns a string of all users data from the users file.
1564
1565 -- THIS FUNCTION NEEDS TO CHECK THAT DATA IS VALID
1566
1567 on readUsersFile
1568
1569     -- Set up where to find the users file
1570     put the pathName & "users" into whatFile
1571
1572     -- Start up Fileio Xtra
1573     set instanceOfXtra = new(xtra "fileio")
1574
1575     -- Set up Fileio to read from users file
1576     openFile(instanceOfXtra, whatFile, 1)
1577
1578
1579     -- If file users doesn't exist, create it
1580
1581     if status(instanceOfXtra) <> 0 then
1582         createFile(instanceOfXtra, whatFile)
1583         openFile(instanceOfXtra, whatFile, 1)
1584     end if
1585
1586
1587     -- Read what's currently in the file
1588     set whatText = readFile(instanceOfXtra)
1589
1590
1591     -- if no users are defined, assume administrator as default user
1592     -- Administrator info is not written into the user's file until at
1593     -- least one user is defined. This occurs in AddUsers functions.
1594
1595     if whatText = "" then
1596         -- for now, assume admin has mail file in each
1597         -- location where kidcode is installed
1598         put "administrator,kidcode,0," & the pathName & "administrator" &
1599         RETURN into whatText
1600     end if
1601
```

Appendix A: KidCode® Lingo Client/Server Email Main Scripts

Page 31

```
1602      -- Close Fileio Xtra
1603      closeFile(instanceOfXtra)
1604      set instanceOfXtra = 0
1605
1606      return(whatText) -- string read from users file
1607
1608  end readUsersFile
1609
1610
1611  -----
1612  -- more users file scripts
1613
1614  on writeUsersFile
1615      global emG_registeredUsers, emG_passwordList, emG_userGroupList,
1616      emG_mailFileList
1617
1618      -- Set up where to find the users file
1619      put the pathName & "users" into whatFile
1620
1621      -- Start up Fileio Xtra
1622      set instanceOfXtra = new(xtra "fileio")
1623
1624      -- Set up Fileio to read and write from/to users file
1625      openFile(instanceOfXtra, whatFile, 0)
1626
1627      -- If file users doesn't exist, create it and set it up for
1628      read/write
1629
1630      if status(instanceOfXtra) <> 0 then
1631          createFile(instanceOfXtra, whatFile)
1632          openFile(instanceOfXtra, whatFile, 0)
1633      end if
1634
1635      -- Put the cursor at the begining of the users file
1636      setPosition(instanceOfXtra, 0)
1637
1638      --- put together string of usersData
1639      set whatText = ""
1640      repeat with uname in emG_registeredUsers
1641
1642          set pw = getProp(emG_passwordList, uname)
1643          set ugroup = getProp(emG_userGroupList, uname)
1644          set mfile = getProp(emG_mailFileList, uname)
1645          set whatText = whatText & uname & "," & pw & "," & ugroup & "," &
1646      mfile & RETURN
1647
1648      end repeat
1649
1650      -- Overwrite users file with updated list
1651      writeString(instanceOfXtra, whatText)
1652
1653      -- Close Fileio Xtra
1654
1655      closeFile(instanceOfXtra)
```

Appendix A: KidCode@ Lingo Client/Server Email Main Scripts

Page 32

```
1656
1657     set instanceOfXtra = 0
1658     return(1)
1659
1660 end writeUsersFile
1661
1662 -----
1663 --- these next functions are created to do file checking
1664 --- however they appear to suffer from severe crash problems
1665 --- these problems will also effect mail file creation if
1666 --- path names are invalid...we need to fix this
1667
1668 on pathp pathname
1669
1670     set instanceOfXtra = new(xtra "fileio")
1671     openFile(instanceOfXtra, pathname, 1)
1672     set theval = status(instanceOfXtra)
1673
1674     case theval of
1675         0 :
1676             closeFile(instanceOfXtra)
1677             set instanceOfXtra = 0
1678             return(1)
1679
1680         -36: -- I/O Error...likely to cause system crash
1681             alert "System has become unstable. " & RETURN & "Please save your
1682 work."
1683             -- next call to fileio xtra may crash system
1684             set instanceOfXtra = 0
1685             return(0)
1686
1687         otherwise :
1688             alert " " & error(instanceOfXtra, theval)
1689             closeFile(instanceOfXtra)
1690             set instanceOfXtra = 0
1691             return(0)
1692
1693     end case
1694
1695 end pathExists
1696
1697
1698 on foldertest
1699     getNthFileNameInFolder("C:\windows", 1)
1700 end foldertest
```